Image Annotation with Discriminative Model and Annotation Refinement by Visual Similarity Matching

by

Rong Hu

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2009

Author ....
Department of Electrical Engineering and Computer Science
August 21, 2009

Certified by.. [illegible] ....
Edward Chang
Director of Research, Google
Thesis Supervisor

Certified by........ [illegible] ....
Berthold Horn
Professor of Computer Science
Thesis Supervisor

Accepted by .... [illegible] ....
Dr. Christopher J. Terman
Chairman, Department Committee on Graduate Theses

Image Annotation with Discriminative Model and Annotation Refinement by Visual Similarity Matching

by

Rong Hu

Submitted to the Department of Electrical Engineering and Computer Science
on August 21, 2009, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Engineering

Abstract

A large percentage of photos on the Internet cannot be reached by search engines because of the absence of textual metadata. Such metadata come from description and tags of the photos by their uploaders. Despite of decades of research, neither model-based and model-free approaches can provide quality annotation to images. In this thesis, I present a hybrid annotation pipeline that combines both approaches in hopes of increasing the accuracy of the resulting annotations. Given an unlabeled image, the first step is to suggest some words via a trained model optimized for retrieval of images from text. Though the trained model cannot always provide highly relevant words, they can be used as initial keywords to query a large web image repository and obtain text associated with retrieved images. We then use perceptual features (e.g., color, texture, shape, and local characteristics) to match the retrieved images with the query photo and use visual similarity to rank the relevance of suggested annotations for the query photo.

Thesis Supervisor: Edward Chang
Title: Director of Research, Google

Thesis Supervisor: Berthold Horn
Title: Professor of Computer Science

Acknowledgments

I would like to thank my thesis advisor Berthold Horn, an excellent advisor who provided insightful comments and constructive criticism for my thesis drafts, and who has been very patient with my progress. I am very lucky to be able to benefit from his expertise in computer vision.

I would like to thank my supervisors and coworkers at Google Beijing for the opportunity to work on this exciting project. Many thanks to Edward Chang for bringing me onboard. His many years of experience in the field of machine learning and multimedia systems was invaluable to this project. Kudos to Yi Wang, for introducing me to the image annotation team and for interesting discussions of machine learning algorithms such as Latent Dirichlet Allocation. Many thanks to the interns at the Beijing office who answered my coding questions and kept life fun and interesting in China.

A huge thank you goes to the people in the image annotation team: Jiakai Liu, Bo Tao, and Meihong Wang. Their ideas, insightful discussions, and hard work pushed this project forward. As did their excellent engineering skills and tireless efforts to connect with researchers at other offices. I will always remember the late nights we spent working together and the early morning video conferences across multiple time zones.

Many thanks to researchers at other Google offices who provided help on this project. Thanks to Samy Bengio and Jingbin Wang for their valuable suggestions and the use of their code. This project would've been impossible without the huge existing codebase and computing infrastructure at Google.

Finally, I owe so much to my friends and family who have supported me through the five years at MIT. I would like to thank Tina Wen, who has set an example for me of high standards of excellence, hard work, and resourcefulness. To Ceryen Tan and Alvin Cheung, who kept me sane over the summer spent writing this thesis. To my parents, Ling and Yu, without whose hard work and sacrifices I would not be here today.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Due to the popularity of digital cameras and photo-sharing sites, the number of images on the Internet is growing at a staggering pace. Users of photo-sharing sites such as Flickr [1] and Picasaweb [2] have uploaded billions of pictures. However, the vast majority of images publicly available on these sites have little or no textual metadata. The kind of metadata that matter for search purposes are usually not those provided automatically by the camera, such as shutter speed and aperture settings, but manual annotations that tell the story of the photo. Examples of such metadata include title, caption, tags, and other words found close to an image in the layout of the containing web page. These words provide clues about the semantics of a picture and can be used to index an image for retrieval.

The lack of good textual annotations makes it difficult to effectively organize pictures on the web, because most search engines only allow their users to search for images by keywords. A typical search engine of web pages first builds an index that maps words to pages containing those words, much like the index at the end of a textbook. When a user inputs a string of query keywords, web pages containing a subset of the keywords are retrieved and ranked. Similarly, a typical image search engine indexes pictures on the web by textual metadata (filenames, surrounding text, or tags) and usually completely ignores the visual content of the pictures. When a

[1] http://www.flickr.com
[2] http://picasaweb.google.com

user enters a search query, those images in the index whose surrounding text contains a subset of the query words are retrieved and ranked. When a picture has little or no textual metadata, as in the case of Picasaweb and Flickr photos, it is effectively impossible to find.

Content-based image retrieval (CBIR) systems approach the problem of finding desired images by providing alternative ways for people to search. Instead of entering text queries, an user uploads a picture, a region of a picture, or a group of pictures as query. CBIR systems index images not by text but by visual features such as color, texture, and interest points (corners and edges). The system processes a query image by extracting visual features from the pixels and returns pictures from its index that have matching features. CBIR derives its name from the fact that image content is processed to aid in retrieval. Figure 1-1 shows the interface of a query by example system developed by Stanford researchers, SIMPLIcity [27]. The example query is a picture of a dog standing in a grassy field. The search results show visually similar pictures that also contain grass and animals. However, the main object of the query, a dog, is missing from most of the results. This example illustrates one of the challenges to retrieval based on visual characteristics: the query maybe ambiguous about what is its most prominent object. In this case a refined query, perhaps the outline of the region containing just the dog, may yield better results. But maybe the user desire not just any kind of dog but exactly the border collie captured in the query. In this case the problem with the search results is harder to fix, due to the wide variation among objects of the same kind. Also, while CBIR is great when users already have image files available or are capable at sketching what they want, the most prevalent and convenient way to search remains that of using textual keywords. It is therefore very useful to have a method to efficiently annotate a large number of web images with reasonable keywords. This is the problem that automatic image annotation tries to address.

Automatic image annotation (or image annotation for short) is the computation of textual metadata for images (i.e. captions and keywords). It involves the discovery of mappings between the visual features of pictures and textual words, usually through

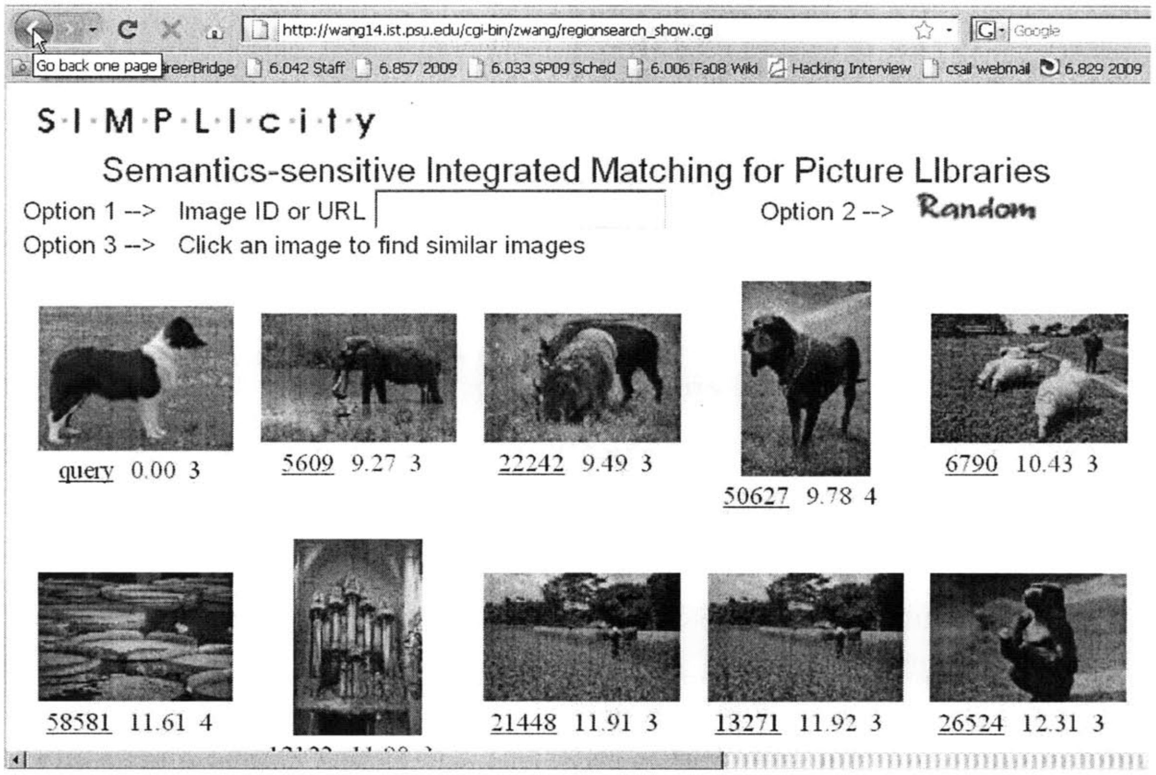

http://wang14.ist.psu.edu/cgi-bin/zwang/regionsearch_show.cgi
Go back one page
S·I·M·P·L·I·c·i·t·y
Semantics-sensitive Integrated Matching for Picture LIbraries
Option 1 --> Image ID or URL
Option 2 --> Random
Option 3 --> Click an image to find similar images
query 0.00 3
5609 9.27 3
22242 9.49 3
50627 9.78 4
6790 10.43 3
58581 11.61 4
21448 11.91 3
13271 11.92 3
26524 12.31 3

Figure 1-1: SIMPLIcity: a query-by-example image retrieval system

machine learning techniques such as classification and probabilistic modeling. Both image annotation as well as content-based image retrieval have been researched for over a decade. But due to the large semantic gap [23], advances in this area have been limited. Effectiveness of techniques depend largely on the corpora. In a narrow domain of hundreds or a few thousands of classes, challenges of image annotation lie mainly in building good models that incorporate domain knowledge and generalize well on new data. In a broad domain such as the Internet, where image properties and subjects vary considerably across the spectrum, annotation methods benefit more from searching in the visual feature space on a large scale. Instead of choosing between a purely model-based and or a purely retrieval-based approach to annotation, we investigate a pipeline that integrate the two approaches in order to take advantage of the benefits offered by both.

1.1 Overview of Annotation System

Figure 1-2 shows a diagram of our hybrid annotation framework. There are four stages in this framework, which will be explained in more detail in the rest of this thesis:

1. **Generate rough initial annotations**: Train a discriminative retrieval model on a large corpus of images with reasonable text. The model we trained was a linear SVM-like model called PAMIR [12]. To label a new picture, represent the picture by a vector of features, use the trained weight vector to project the image from its visual feature space to a term weights vector in the textual space. Then take the top weighted words from the text vector as crude initial seed words.

2. **Text-based retrieval**: The initial annotations are used to retrieve semantically-related pictures from Google image search. Google image search retrieves up to 1000 results per query, therefore the total number of distinct images in this set is determined in part by the number of initial annotations generated.

3. **Content-based matching**: Extract features from the retrieved images and build a database of feature descriptors for matching. The type of features used in this step does not have to be the same as in step 1. Run approximate nearest-neighbors search on the database of feature vectors to find images most visually similar to the unlabeled picture and output with each matching image similarity scores.

4. **Annotation refinement by label transfer**: Aggregate keywords from visually similar images and use these to refine and rerank the original set of annotations.

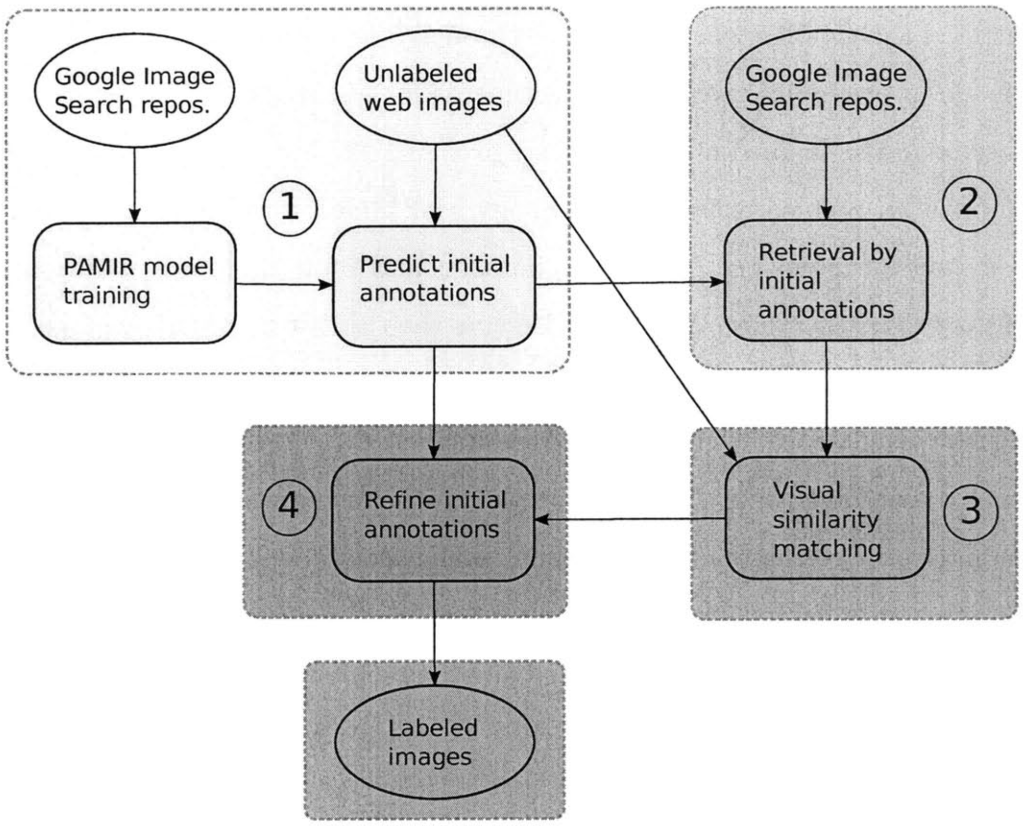

Google Image Search repos.
Unlabeled web images
1
PAMIR model training
Predict initial annotations
Google Image Search repos.
2
Retrieval by initial annotations
4
Refine initial annotations
Visual similarity matching
3
Labeled images

Figure 1-2: Overview of our image annotation pipeline.
1) First we train a discriminative model optimized for image retrieval by text. The training data is derived from Google Image Search repository. Given an unlabeled image, this model can suggest some rough initial keywords.
2) These keywords are used to retrieve a small set of images from Google Image Search, some of which will be semantically related to the unlabeled photo.
3) Nearest-neighbor search for visual similarity matching is done on the retrieved set to find images similar to the unlabeled image.
4) Labels from similar images are processed to refine the initial annotations.

1.2 Thesis Outline

In this thesis, I describe a hybrid approach for annotating images that combines both a trained model and content-based retrieval for annotation. Chapter 2 covers background and related work in image annotation helpful for understanding the development of this new system. Chapters 3 through 5 describe in detail each important stage in the process and associated experiments for evaluation of that stage. Chapter 3 discuss model trained in the initial stage: PAMIR (Passive-Aggressive Model for Image Retrieval), which was originally created for image search result re-ranking. It includes some statistics on the training datasets and shows how size of training corpus affects model accuracy. Chapter 4 presents the third stage of the pipeline: visual similarity matching on a relatively small set of images retrieved by initial annotation guesses, the visual features chosen and experiments with the features to see how they perform for matching tasks. Chapter 5 explains how to aggregate tags from matching images and vote/weight the tags by visual similarity scores. It also presents overall annotation results that show some improvements to the initial annotations. Finally, Chapter 6 summarizes the lessons learned and lists some areas for further exploration in future works.

1.3 Contributions

While there have been many methods for processing images, feature extraction, and learning patterns for associating text and images, designing an annotation system involves making many choices, such as which methods and features to use, how to combine the different methods, how to evaluate the system at each step and tune the parameters. These choices should be guided by an analysis of the trade-offs. Although this thesis draws heavily upon the works of other researchers, it does make the following contributions:

1. A new way of combining model-based and search-based methods together into an annotation pipeline is introduced. Although this thesis presents a specific

model and two specific kinds of features for retrieval, these models and features can be changed for others easily. It is the pipeline itself: the order that these components are put together that is novel.

2. This thesis presents methods for tuning the system and evaluating it at each stage, such as using the coverage rate to choose the number of initial annotations to output, and using the cumulative number of correct annotations to evaluate the performance of annotation re-ranking. Separate evaluations of each stage is important for system tuning because it helps to find those places that have the most influence on the overall annotation quality.

3. Transfering text from visually similar images has not been investigated much before. This thesis introduces a method for annotation refinement using information found in the search query log. The search query log contains feedback from users, making its association of image and text more reliable than the surrounding text. To the best of my knowledge, this source of text has not been considered in any annotation system before.

Chapter 2

Background and Related Work

There are two ways to view automatic image annotation. The traditional view treats annotation as a machine learning problem, with two modeling approaches under this view: *discriminative* and *generative*. The discriminative, or supervised approach treats annotation as a classification problem, where each label word is a class representing either a semantic concept or a specific object. The simplest method is to train a series of binary classifiers, one for each label, as done by Goh et al. [11]. This approach is quite successful in limited domains: to detect faces [22], horses [9], to distinguish cities from landscapes [25] and so forth. Other methods in this camp focus on multi-label classification [4], where labels are correlated classes and each image can be assigned more than one class, and training for all classes are done together at the same time.

The generative approach relies on probabilistic models to learn the co-occurence relationship between image features and semantic labels. Duygulu et al. [5] adopted a machine translation model to translate image blobs into label words. Jeon et al. [13] used a cross-media relevance model (CMRM), which assumes label words and blobs are conditionally independent given an image. These early works inspired several other variations such as Continuous-Space Relevance Model (CRM) [15], Dual-CMRM [18], and Multiple Bernoulli Relevance Model (MBRM) [7]. Li et al. built the ALIP system [16] by training a Multiresolution Hidden Markov Model for every semantic concept. At the same time, latent space models from text processing, such as Latent Semantic Analysis (LSA), Probablistic Latent Semantic Analysis (PLSA), and Correspondence Latent Dirichlet Allocation (Corr-LDA) have all been applied to image annotation [20, 1, 3] with varying success. Liu et al. trained the original LDA model on images represented as discrete bags of words and used the model to infer the most likely annotations [17]. Because the training time and space usually grow rapidly with vocabulary size, most model-based approaches limit the vocabulary size to a few hundred. To the best of our knowledge, the biggest vocabulary has 950 words and is taken by Carneiro et al. from the Corel 30K stock photos.

While the traditional view of automatic image annotation leans heavily on model building, recent works have shifted toward data-driven methods [29, 28, 24]. Given an unlabeled image, the idea of data-driven annotation is to search the Web for a set of visually similar images. Web images usually have a lot of surrounding text, so they can be processed and the text transferred as annotations for the unlabeled image. Although conceptually simple and straightforward, purely data-driven methods can achieve surprisingly good performance. Torralba et al. [24] compiled a database of 80 million 32×32 tiny Web images and demonstrated that using k-nearest neighbor search, a simple voting scheme, and a WordNet dictionary, the performance for person detection is comparable to the popular Viola-Jones face detector. However, the data-centric approach encounters two problems. First, it encounters the "semantic gap" [23] problem head on. Because visually similar images need not be semantically related, the choice of image features and similarity measure greatly influence the results. The second problem is scalability. It is non-trivial to search huge collections of images due to the high-dimensionality of images. Because it is impossible to store billions of images in memory, accuracy usually becomes the trade-off. The feature vectors is often reduced by principal components analysis (PCA) to less than 50 and in some cases reduced further by hashing [28]. Because information is discarded in the process of compression, the performance of nearest neighbor search also suffers. The study by Torralba et al. [24] shows that as the dataset grows in size, the number of *approximate* nearest neighbors we must examine to guarantee a high probability of getting a fixed number of *exact* nearest neighbors also grows. For a dataset of 79 million images, $16,000$ *approximate* nearest neighbors must be considered to achieve $> 80\%$ probability of including 50 *exact* nearest neighbors. As the dataset grows, retrieval recall rate also suffers tremendously.

To overcome the scalability problem, researchers have tried to reduce set of images on which to do visual similarity matching. Yeh et al. [29] reduce the set by restricting the domain to university landmark recognition in mobile images. They first perform content-based search on a small bootstrap set of $2,000+$ landmark images and extract keywords associated with matched images. Next they perform text search using the extracted keywords to retrieve a set of images from Google. Finally they cluster the image results visually to obtain additional words. In [28], Wang et al. *assume* at least one keyword is associated with a given image (e.g. folder name) and use that keyword to retrieve semantically related images from Google. The search results are clustered and one keyword per cluster is generated as annotation. However, the solution to finding adequate initial keywords is left as future work.

Chapter 3

PAMIR

The *Passive-Aggressive Model for Image Annotation* (PAMIR) [12] was developed by David Grangier and Samy Bengio. It is a model optimized for the *ranking* of images retrieved by textual queries. Inspired by the Ranking SVM [14] for web page re-ranking, PAMIR trains a maximum-margin classifier that separates relevant pictures from irrelevant pictures for each query, so that when the given query is performed, all relevant pictures rank higher than all irrelevant pictures. It finds the classifier with the largest margin in order to achieve good generalization performance on unseen queries. In addition, it uses an efficient online Passive-Aggressive algorithm for training, which has a time complexity that is linear in the number of training instances as opposed to standard SVM training, which has a time complexity that is at least square in the number of training instances. On the Corel dataset, a standard benchmark for image annotation and retrieval, PAMIR was shown to outperform models such as CMRM, PLSA, and SVM.

The following sections describe PAMIR in detail, and how we use it as the first stage in the annotation pipeline. The purpose of this stage is to generate a rough set of initial annotations per picture. While many words in this initial set can be completely unrelated to the picture, hopefully the model is good enough that the initial set contains words that an average person would consider descriptive of the picture semantics. Then retrieval with these initial words as queries would return semantically related images, which can be used to refine the initial annotations.

3.1 Image representation

PAMIR's approach to image representation is similar to most other annotation systems: each picture is represented as a bag of local descriptors. First, a picture is divided into overlapping 64×64 pixel blocks. Smaller images may have fewer blocks and an upper limit of 77 blocks per image is enforced. Then, a Local Binary Patterns (LBP) [21] histogram and a color histogram are extracted from each block, concatenated to form one block descriptor. The color histogram is obtained by first building an RGB color codebook. The number of bins of the histogram is the size of the color codebook, and each pixel is binned to the closest codebook color. Local Binary Pattern is a simple yet powerful local texture descriptor that is gray scale and rotation invariant. The pattern used in PAMIR is $LBP_{8,2}$, an example of which is shown in Figure 3-1. For each pixel in the block, its intensity level is compared with 8 neighboring pixels spread evenly on a circle of radius 2. Results of the 8 comparisons form an 8-bit binary sequence starting with the neighbor directly to the right of the center pixel. A 1 in the sequence represents the case where the neighbor's intensity is higher (the pixel appears lighter) than the center pixel and a 0 otherwise. Since there are $2^8 = 256$ possible 8-bit sequences, a LBP histogram would naturally have 256 bins. However, sequences like 11000001 and 00000111 can actually be derived from the same pixel in the same picture simply by rotating the picture. To achieve rotation invariance, some 8-bit sequences can be considered equivalent and grouped into the same bin. Indeed, Ojala et. al. [21] showed that sequences with more than two 0/1 transitions can be put into one bin with little change in performance. In PAMIR, the total number of bins in the LBP histogram is reduced to 59.

As mentioned before, the LBP histogram and color histogram are concatenated to form one descriptor per block, and each image contains many blocks (up to 77). The next step involves discretizing the block descriptors from all training images into a codebook of "visterms", or visual words, using k-means clustering. While this "visterm kernel" approach is not the only way to represent images, it is shown in [12] to outperform many other picture kernels for PAMIR training, and so we do

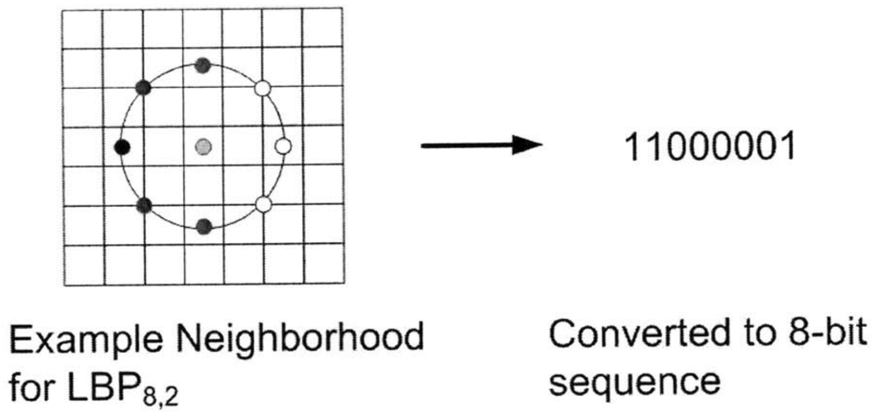

11000001
Example Neighborhood for $LBP_{8,2}$
Converted to 8-bit sequence

Figure 3-1: $LBP_{8,2}$ example. The gray scale intensity of the center pixel is compared with the intensities of its 8 neighbors at a distance 2 pixels away and result is converted to an 8-bit binary sequence

not consider the others here. Once we have a codebook or dictionary of visual words, each picture can now be represented as a bag of words just like a text document. More concretely, we can represent a picture as a sparse vector $\mathbf{p} \in \mathbb{R}^C$, where C is the size of the codebook. Component i of $\mathbf{p}$ represents the i^{th} visterm in the codebook. The value of component i is the normalized term frequency-inverse document frequency (tf-idf) of the i^{th} visterm. Tf-idf is a measure widely used in text-mining to assign an importance weight to a word in a document. It is defined as the following:

$$\mathbf{p}_i = \frac{\text{tf}_{\mathbf{p},i} \cdot \text{idf}_i}{\sqrt{\sum_{k=1}^{C} (\text{tf}_{\mathbf{p},k} \cdot \text{idf}_k)^2}} \tag{3.1}$$

The term frequency ($\text{tf}_{\mathbf{p},i}$) is the the number of times the i^{th} visual word appears in the picture $\mathbf{p}$. The more times this word appears, the more important it is in describing the picture, and therefore the higher its weight should be. The inverse document frequency (idf_i) is $-log(r_i)$, where r_i is the fraction of training corpus images containing the i^{th} visual word. If a word occurs in a large fraction of training corpus images, then it is not as useful for distinguishing the images apart. An example from text documents is the word "the", which occurs many times in most documents but has very little meaning. Therefore the higher the document frequency, the lower should be the word's weight, which justifies multiplying idf and tf together. Lastly, the visterm vector is normalized to be unit length so that bigger pictures with more blocks and therefore more visterms do not weigh unfairly high in the training process. The bottom of the fraction in Equation 3.1 is the normalization constant.

3.2 Query Representation

Similar to an image, a query is treated as a bag of words and represented also with a sparse term vector. The dimension of the query vector is the size of the textual dictionary, T, as determined by the training corpus. The weight for each term in the vector is similar to the normalized tf-idf used for visterm vectors explained in the previous section. But instead of term frequency, a binary value indicating the word's presence or absence in the query is multiplied with the idf. This is because in natural language, if a word appears multiple times in a query, it usually doesn't change the meaning of the query too much; but for pictures, the repetition of a patch will likely yield a very different interpretation, so term frequency should be kept to incorporate as much information as possible.

3.3 Model Training

Given a query represented by a term vector $\mathbf{q} \in \mathbb{R}^{\mathbf{T}}$, a set of relevant pictures $\mathbf{p}^+$ and irrelevant pictures $\mathbf{p}^- \in \mathbb{R}^{\mathbf{C}}$, the goal of PAMIR is to train a scoring function F_w so that all relevant images are ranked higher than all irrelevant images. The goal is expressed by the following equation:

$$\forall \mathbf{p}^+, \mathbf{p}^-, F_w(\mathbf{q}, \mathbf{p}^+) > F_w(\mathbf{q}, \mathbf{p}^-) \tag{3.2}$$

In the simplest case, PAMIR trains a linear model W, a $T \times P$ matrix of parameters, and uses the product $\mathbf{q}^\mathbf{T}\mathbf{W}\mathbf{p}$ as the relevance score:

$$F_w(\mathbf{q},\mathbf{p}) = \begin{bmatrix} q_1 & q_2 & \cdots & q_T \end{bmatrix} \underbrace{\begin{bmatrix} - & \mathbf{w_1} & - \\ & \vdots & \\ - & \mathbf{w_T} & - \end{bmatrix}}_{\mathbf{W}} \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_C \end{bmatrix} \tag{3.3}$$

From Equation 3.3, we can see that the matrix of model parameters W projects a picture vector from the visual feature space into the textual space. The relevance score then is just the inner product between the projected picture and query.

To understand training better, it helps to rewrite W by concatenating its rows together into a single row vector of length TP:

$$w = \begin{bmatrix} \mathbf{w_1} & \mathbf{w_2} & \cdots & \mathbf{w_T} \end{bmatrix}$$

Similarly,we can combine a query vector $\mathbf{q}$ and a picture vector $\mathbf{p}$ into a single row vector $\gamma(\mathbf{q},\mathbf{p})$ of length TP:

$$\gamma(\mathbf{q},\mathbf{p}) = \begin{bmatrix} q_1\mathbf{p} & q_2\mathbf{p} & \cdots & q_T\mathbf{p} \end{bmatrix}$$

One can verify that the dot product of the new row vectors, $w \cdot \gamma(\mathbf{q},\mathbf{p})$, is exactly the relevance score $F_w(\mathbf{q},\mathbf{p})$. Written this way, the relevance score can be interpreted as the projection of $\gamma(\mathbf{q},\mathbf{p})$ on w. The difference between the scores of two pictures $\mathbf{p_1}$ and $\mathbf{p_2}$ can be simply written as $w \cdot \gamma(\mathbf{q},\Delta\mathbf{p})$, where $\Delta\mathbf{p} = \mathbf{p_1} - \mathbf{p_2}$.

Each training data point of PAMIR is a triple $(\mathbf{q},\mathbf{p}^+,\mathbf{p}^-)$. Finding the w that satisfy Equation 3.2 with the maximum margin, while also allowing for overlapping class distributions is equivalent to solving the following optimization problem:

$$\min_w \quad \|w\|^2 \quad + \quad C\sum_{i=1}^{M}\xi_n \tag{3.4}$$

with the constraints $\forall i, 1 \leq i \leq M$:

$$\begin{cases} w \cdot \gamma(\mathbf{q}_i, \Delta\mathbf{p}_i) \geq 1 - \xi_i, & \Delta\mathbf{p}_i = \mathbf{p}_i^+ - \mathbf{p}_i^- \\ \xi_i \geq 0 \end{cases}$$

where M is the total number of training instances; ξ_i are slack variables that allow for some training data points to fall within the margin or misclassified; and C controls the trade-off between the accuracy on the training set and the margin width. The training algorithm starts with $w^0 = 0$ and goes through n iterations as specified by the user. In the i^{th} iteration, w^i is updated with the following equations:

$$w^i = w^{i-1} + \tau_i v_i \tag{3.5}$$

$$v_i = \gamma(\mathbf{q}_i, \Delta\mathbf{p}_i) \tag{3.6}$$

$$\tau_i = \min\left\{c, \quad \frac{\max(0, \quad 1 - w^{i-1} \cdot v_i)}{||v_i||^2}\right\} \tag{3.7}$$

In each iteration, the amount by which the weight vector changes depends on the aggressiveness parameter c and the margin of the i^{th} training instances with the previous weight vector. The number of iterations controls the fit. Setting the value of n too high will likely result in overfitting and poor generalization on test data while setting it too low may not get the best classifier. Both parameters c and n are found by cross-validation, a way to estimate model performance and reduce chances of overfitting. For cross-validation, the training corpus is divided into a model construction set and a validation set. The model is trained using the model construction set and tested on the validation set. To tune the parameters, we train many models using different combinations of paremeter values (c, n) on the model construction set. The combination of parameter values that give the lowest error on the validation set are chosen and the model is retrained using the entire training corpus and the chosen parameters. To ensure that the validation set and the model construction set have similar properties, training corpus points must be partitioned randomly (e.g. a coin flip to determine to which set a point belongs).

3.4 Training Corpus

PAMIR's training algorithm takes as inputs triples of the form $(\mathbf{q}, \mathbf{p}^+, \mathbf{p}^-)$. These triples actually come from a very sparse binary relevance matrix R with rows representing queries and columns corresponding to pictures, where $R_{i,j} = 1$ if picture j is relevant to query i and $R_{i,j} = 0$ otherwise. The training data derives from Google image search's query logs, which contains for each query, a list of retrieved pictures and the number of clicks each picture got within a window of time.

A picture can be deemed relevant to a query if its click number exceeds a certain threshold, which may be different for each query due to varying levels of query popularity. Alternatively, the percentage of total clicks instead of actual number may be more accurate. One factor that makes the query logs noisy is that users seldom look at every item in the search result, so pictures listed in the first few result pages are much more likely to get clicks than pictures listed in later pages, regardless of actual relevance, simply because few people look at the later picture. This is complicated by the fact that some queries return hundreds of pages of results all of which contain relevant pictures, while others return less than ten pages only a couple of which contain anything relevant. Given clickthrough data aggregated across millions of users, it is hard to determine for a picture with few clicks whether the dominant cause was its lower ranking or irrelevance. While high quality data for training does influence the outcome, it is not a focus of this project. Fortunately, the PAMIR learning algorithm can handle noisy nonlinearly separable training data natively to some extent with the slack variables.

After processing the clickthrough logs to retain only English queries and safe search content (i.e. eliminating queries and results for nude pictures), the training corpus consist of 2.2 million images from the search results of 140K queries. The dictionary for the queries contains 21K stemmed words.

3.5 Computing and Evaluating Initial Annotations

Once we train a PAMIR model W, we can project any picture from the visual space $\mathbf{p} \in \mathbb{R}^{\mathbf{C}}$ into a term weights vector $\mathbf{q}' = W\mathbf{p}$ in the textual space. The simplest way to make use of $\mathbf{q}'$ for annotation is to take the top N words with the highest weights in $\mathbf{q}'$ as initial annotations for the picture, since these terms contribute the most to the relevance score of a picture.

The parameter N should be chosen with care since the next stages of the pipeline will use these words to retrieve $1000N$ images from Google image search and perform visual similarity matching on the retrieved images. If N is too small, then none of the initial annotations may be relevant to the picture, and retrieval for semantically related photos is mostly useless. If N is big, several hundred for example, then a picture is highly likely to get at least one relevant annotation. But the larger N is, the slower and more work needed to refine annotations by similarity matching later. Furthermore, similarity matching depends on a visual features index that must fit entirely inside a computer's memory during matching. There is an upper limit to the amount of memory on one commodity machine, and by extension there is an upper limit on the value of of N, which is on the order of a few tens. Therefore, finding a good value for N involves making a trade-off between the accuracy of initial annotations and the accuracy of the visual matching stage.

To choose a reasonable N, we randomly sampled 50 public photos from Picasaweb albums and 50 photos from the Corel 5K dataset. Since the two sets are small and the Picasaweb photos have no ground truth labels, we manually examined the top 20 annotations for each photo, and recorded the number of photos that have at least one good annotation. We define the *coverage rate* to be the percent of unlabeled image having at least one good annotation generated by PAMIR. A curve of the coverage rate against the number of predicted annotations is shown below in Figure 3-2.

The curve is convex: rising quickly at first and flattens as the number of annotations increases. The point that seems optimal is around 11. Beyond the top 11 words, the coverage rate improves only marginally with additional words, at the cost

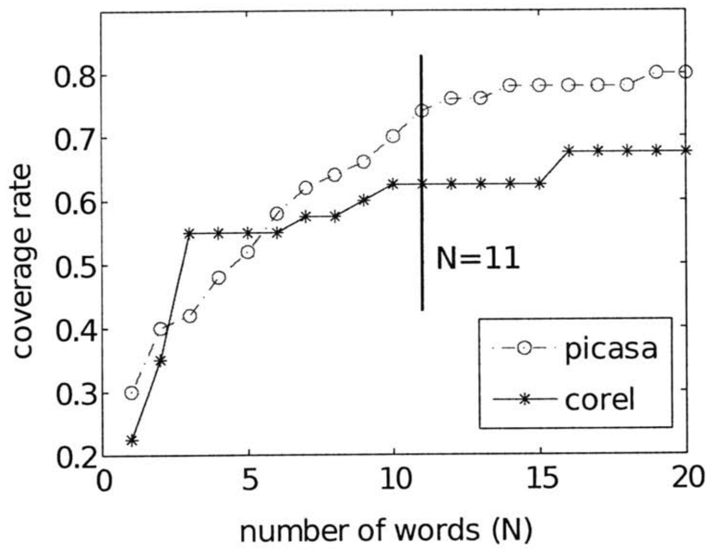


Figure 3-2: PAMIR initial annotation coverage rates for Picasaweb and Corel photos.

of more resources for similarity computation later. We choose to output 11 initial annotations for all later experiments. Because the sample size is very small and we evaluated correctness manually, the coverage rate is an overestimate of the actual coverage rate if we were able to calculate it on the entire Picasaweb and Corel 5K datasets. At $N = 11$, it is 74% for the Picasaweb sample and 63% for the Corel sample.

3.5.1 Evaluation Metrics

We use two metrics to evaluate the quality of initial annotation, listed in Table 3.1 for convenience. One metric is the coverage rate as mentioned before. We believe that at this stage in the pipeline, the coverage rate is a reasonable measure of quality, because in the next stage these initial seed words help to retrieve more images *semantically* similar to the unlabeled picture. If we can produce at least one relevant initial seed word for a large number of pictures, then retrieval and refinement steps are more likely to be useful.

To compare the quality of the initial annotations against that of the final annotations, a second metric is needed: the number of correct annotations is a straightforward choice. Evaluation of correctness can be done either completely manually

Evaluation metrics	**Description**
coverage rate	Percent of pictures with at least one correct annotation
avrg. no. correct annotations	Average number of correct annotations per image for those images with at least one good annotation
total no. correct annotations	Total number of correct annotations on the test set

Table 3.1: Annotation quality metrics

or automatically on a test set with manual ground truth annotations. Whether the refinement stages are useful can be measured by the amount of increase in the number of correct annotations from the initial stage.

To better evaluate the quality of initial annotations, we selected a larger dataset of web images with quality ground truth annotations. This dataset consist of 1,617 photos from the Google Image Labeler game [1]. The Google Image Labeler is a version of the ESP game [2] licensed by Google, which harnesses the power of humans to label images. The game matches two random players as partners and shows them a series of photos. The players cannot communicate other than keep typing words to describe the current image. Only when they both agree on the same label for it can they get points and move on to the next image. This setup ensures that players type reasonable tags for each picture. The game draws pictures from the Google Image Search repository. We filter the Image Labeler dataset further by retaining only pictures having at least 30 tags, resulting in 1,617 pictures. This will be refered to as the Image Labeler dataset, although it represent only a tiny portion of the complete Labeler dataset. We consider *all* words associated with a Labeler image to be ground truth, even if they have not been matched on by any pair of players, because they are produced by humans.

On this Image Labeler dataset, the coverage rate and average number of correct annotations are computed automatically. We stemmed both PAMIR and the Labeler

[1] http://images.google.com/imagelabeler/

[2] http://www.espgame.org/gwap/, for more information, see [26]

words before matching them. This takes care of plurals and ending changes for most words. A PAMIR annotation is considered correct if it matches a ground truth word exactly. We only count exact matches and leave out potential correct annotations from synonyms. This requirement may be a little stringent but it is the simplest to implement. Compared to manual evaluation, this gives a very conservative estimate of annotation quality. We found on this dataset the coverage rate is around 42.5%. Out of the total of $17,787$ words generated by PAMIR, $1,871$ words are correct. The average number of correct annotations for images with at least one correct annotation is 2.5. While this result seems low, it is a lower bound on the actual numbers, and serves as our starting point.

Chapter 4

Visual Similarity Matching

The previous chapter described how to generate some initial annotations of N words with PAMIR for each unlabeled picture $\mathbf{p}_i$. With these annotations as queries, text-based image retrieval is performed to collect a small set of $1000N$ pictures from Google image search, where N is the number of queries and is set to 11 for all experiments. This chapter presents the next step: the processing of the retrieved set to find pictures most visually similar to original picture $\mathbf{p}_i$.

Image matching starts by extracting features from unlabeled pictures and building them into a database of descriptors. The database should use an indexing structure that's efficient for k-nearest neighbor (k-NN) search in high dimensions. Then for each retrieved image r_i, having on the order of several hundred keypoints and associated feature descriptors, feature matching finds the nearest neighbors for all of r_i's features. The pictures in the database with the highest number of matching features are selected for geometric verification, which attempts to find an affine warp from r_i to its matches. If the affine warp results in a large error, the match is discarded. When all retrieved images have been matched this way, we get a relevance matrix that records the similarity scores between unlabeled pictures and retrieved images. This information is used in the next step to refine the initial annotations.

4.1 SIFT

We tried two kinds of features for similarity matching. The first features is the LBP+color histograms used in PAMIR training which was presented in Section 3.1. We tried a second feature for matching: SIFT [19], a distinctive local feature widely used computer vision for matching objects and aligning scenes in images. SIFT features are scale and rotation invariant, so that shrinking the picture doesn't affect feature detection. In addition, SIFT is also partially invariant to illumination and viewpoint changes, affine distortions, occlusion and noise. These properties make SIFT suitable for matching images on the web, which vary greatly in quality, lighting, and viewpoint. This chapter mainly focuses on explaining image similarity matching with SIFT, because matching images with LBP is similar but simpler.

The extraction of SIFT features begins with finding candidate keypoints, which are local maxima/minima in the difference of Gaussian versions (DoG) of the original image at various scales. The original image is converted to gray-scale first, because keypoint detection only searches for peaks in gray-scale intensity values. Generating DoG images involves successively down-sampling the original image by a factor of 2 in both the x and the y directions. The down-sampling method is simple block-averaging: the value of a pixel in the shrunken image is the average of the pixel values in a 2×2 block in the original image. Each down-sampling step produces an octave, in which the original image is convolved with many Gaussians having scales between σ and 2σ. The scale σ is the width of the 2D Gaussian function $G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$. Because a continuous 2D Gaussian function has very small non-zero values covering the entire image, it is not efficient to compute its convolution with the image. In practice a truncated and discretized Gaussian kernel is used for smoothing. The kernel is a square grid with side length $\lceil 6\sigma \rceil$, and the value of each cell in the grid is the value of the continuous Gaussian function sampled at the center of the cell. Table 4.1 shows an example of a 5×5 Gaussian kernel with $\sigma = 0.7746$. Finally, the Gaussian-blurred images of adjacent scales in the same octave are subtracted to produce DoG images. This pyramid process is illustrated

Table 4.1: An example 5×5 Gaussian filter with $\sigma = 0.7746$

0.0003	0.0041	0.0095	0.0041	0.0003
0.0041	0.0502	0.1154	0.0502	0.0041
0.0095	0.1154	0.2656	0.1154	0.0095
0.0041	0.0502	0.1154	0.0502	0.0041
0.0003	0.0041	0.0095	0.0041	0.0003

in Figure 4-1 and an example showing how an image looks like after this process is given in Figure 4-2.

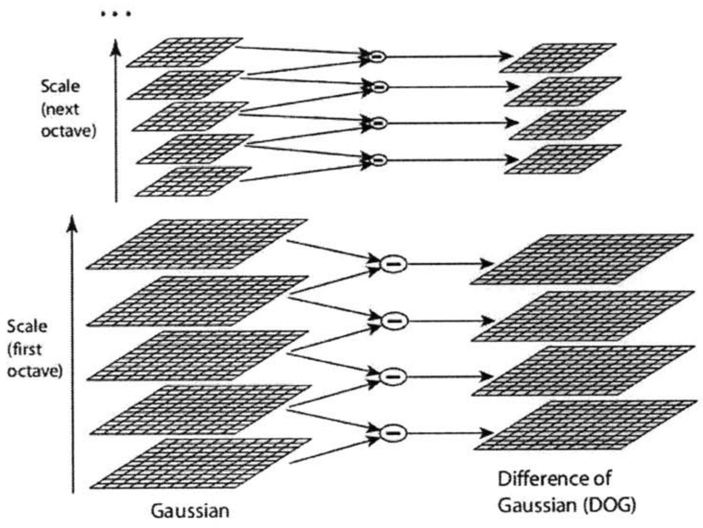


Figure 4-1: A pyramid of difference of Gaussians images for SIFT, taken from Lowe, 2004 [19]

The candidate keypoints are the local extrema of DoG images. They are found by comparing each pixel in the DoG image to its 26 neighboring pixels: 8 neighbors in the current image, and 9 neighbors in each of the two DoG images with adjacent scales. Once a candidate keypoint has been identified, a polynomial fit of its neighborhood intensities is done to determine a more accurate location of the local extrema. This is known as keypoint localization. If the interpolated intensity at this location has a low magnitude (low contrast), the point is discarded because it is unstable. In addition, points lying along edges are also discarded for the same reason. The remaining candidate keypoints are assigned a scale and an orientation. The scale is just the

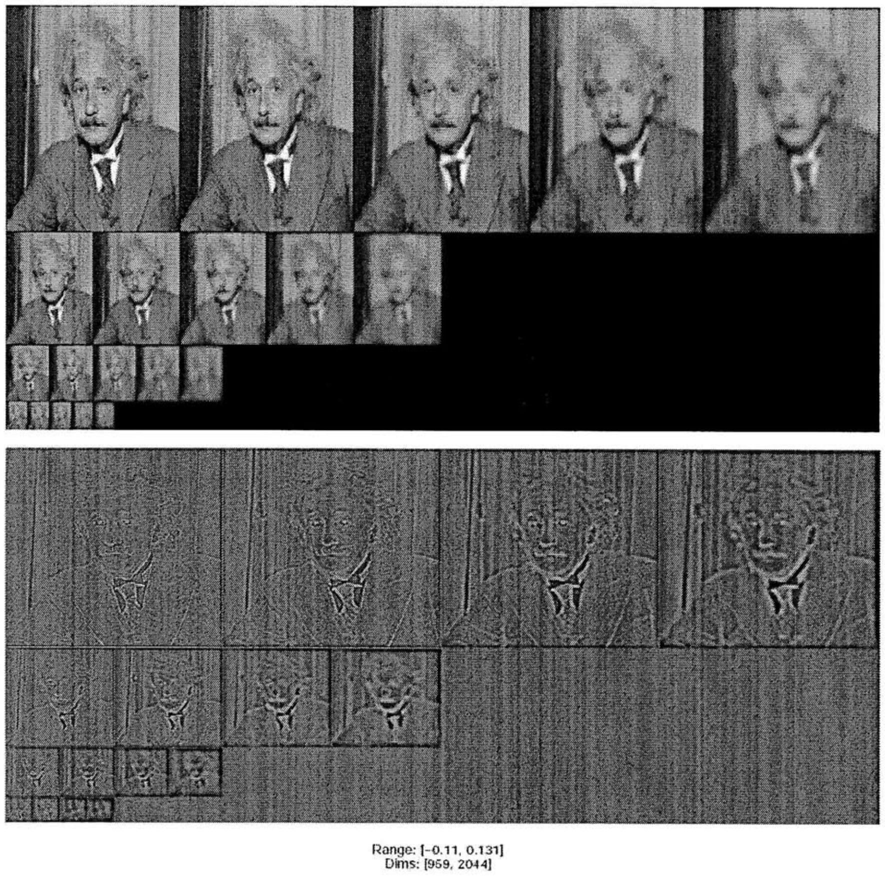


Figure 4-2: An example of Gaussian and difference of Gaussian images, taken from Estrada's SIFT tutorial [6]

scale of the DoG image. The orientation of a keypoint is the dominant direction in a 36-bin histogram of gradients in the keypoint's neighborhood. If more than one directions are dominant, for example, if there are multiple peaks of similar heights in the gradient histogram, another keypoint at the same location is created with the other dominant direction as its orientation.

Finally, a SIFT descriptor is computed for each keypoint. The descriptor is basically a histogram of weighted gradient orientations. The common choice is to use 8 orientations. The descriptor is computed from a local square patch of 16×16 pixels centered at the keypoint and with sides parallel to the keypoint orientation. This patch is smoothed by a Gaussian with σ equal to half of the patch width, so that gradients closer to the center have higher weights. This patch is further divided into 16 blocks of 4×4 pixels, each accumulating an 8-bin histogram. The descriptor is formed by concatenating the 16 histograms together to form a $16 \times 8 = 128$ dimensional vector. To make the descriptor robust against changes in illumination, this vector is normalized to unit-length, and each bin is thresholded to prevent too much influence from any one dimension.

In our system, the SIFT descriptors are further compressed down to 40 dimensions by Principal Component Analysis (PCA). PCA transforms the feature space into a new orthogonal space, where the first dimension contribute the most to the data's variance, followed by the second dimension, and so on. The dimensions that contribute the least to the variance can be discarded because they carry the least amount of information for distinguishing the datapoints apart.

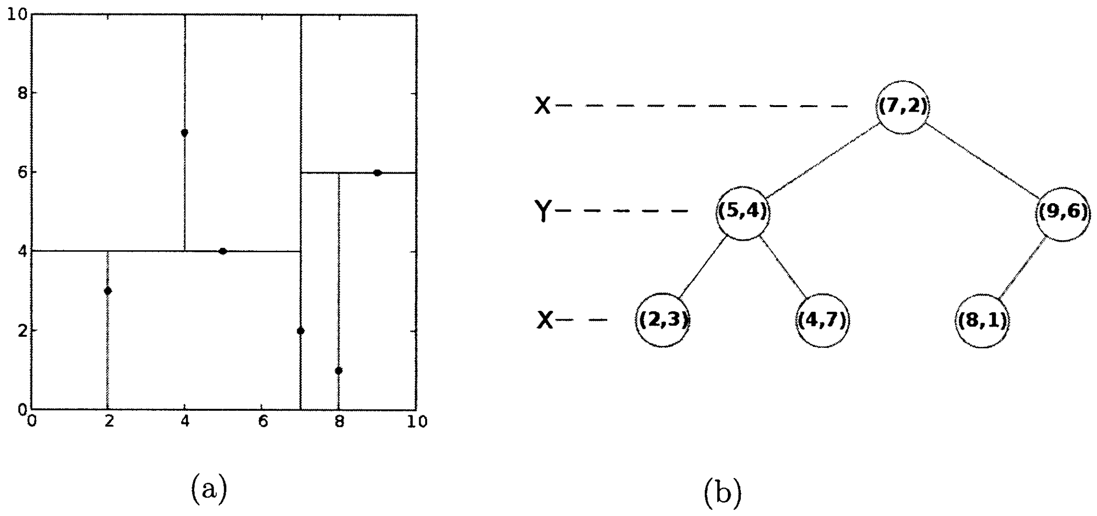
4.2 Feature Index and Matching

After extracting SIFT features from all unlabeled pictures, we store the features descriptors into a database and build an index on this database to facilitate matching. The index data structure we use is the kd-tree, a binary search tree that partitions the feature space with orthogonal hyperplanes. The reason for using the kd-tree is that feature matching relies on nearest neighbor search, and a kd-tree allows much more efficiently NN lookups than brute force linear scan. The distance $d(\mathbf{u}, \mathbf{v})$ between any pair of keypoint descriptors $\mathbf{u}$ and $\mathbf{v}$ for nearest neighbor search is the Euclidean distance: $d(\mathbf{u}, \mathbf{v}) = \sqrt{\sum_{i=1}^{n} \|\mathbf{u}_i - \mathbf{v}_i\|^2}$.

4.2.1 Nearest Neighbor Search on a kd-tree

The kd-tree is built recursively, first by picking a dimension, s, on which to split the datapoints. The standard way is to pick the component of the vector in the direction of highest variance on the current remaining points. Then a pivot point is chosen, which is the datapoint that has the median value m in the splitting dimension. Choosing the pivot to be the median ensures that the kd-tree is balanced. The pivot is made into the root node storing the values s and m, and the remaining points are split evenly in halves, to be turned into subtrees. All datapoints in the left subtree have s-dimension value less than the root, and all points in the right subtree have values greater than the root. The previous steps are repeated in each half, using a different dimension for splitting. This process continues until all points have been inserted into the tree, and takes $O(n \log n)$ time, where n is the number of points. An example of a 2-dimensional kd-tree is showing in Figure 4-3.

Figure 4-3: An example of the kd-tree in two dimensions. (a) Visualizing a kd-tree partitioned space. (b) kd-tree in binary tree form, with dashed lines indicating the splitting plane at each level.



Nearest neighbor search on a kd-tree proceeds by first traversing all the way to the leaf node containing the query point, and saving the distance between the query and this leaf node as the current best distance d_b. The algorithm then backtracks to the parent node and checks if the distance between the query and this node is less than d_b. If it is, the current best is replaced. If not, there may be no points closer than the current best in the other branch, and we can prune that branch. This can be decided by checking whether the hypersphere centered at the query with radius d_b intersects the splitting hyperplane containing the parent node. If it does, we must check the other branch. If not, that branch can be pruned and we can backtrack to a higher level. This procedure is repeated until all points have been examined or pruned.

While the kd-tree is very efficient for *exact* k-NN search in low-dimensions, the number of lookups increases exponentially with the number of data dimensions k, so that the performance is not much better than linear scan when $k > 20$. To shorten the search time, we settle for *approximate* nearest neighbors instead, by using the *Best Bin First* (BBF) search strategy [2]. This strategy limits the number of nodes examined and the order in which they are examined, so that it returns the exact nearest neighbor the vast majority of the times and very close neighbors the rest of the times. It does so with the help of a priority queue, whose size is set by the user (we set it to 200). At each node, the branch not taken is added to the queue along with its distance to the query, which is the minimum distance between the query point and the splitting hyperplane at the node. Upon backtracking, the node with the least distance to the query is examined first. To reduce the number of false matches, the distance ratio between the closest and the second-closest neighbors must be less than a certain threshold (0.8 is the default for SIFT). Matches with a ratio greater than this threshold have a high probability of being false matches.

4.3 Image Matching and Verification with SIFT

For each retrieved image r_i, feature matching traverses the database of keypoint descriptors and finds the nearest neighbor for each keypoint in r_i. Associated with the neighbor is a link to the unlabeled image u_k containing it. The number of links between u_k and r_i is stored in a table and updated as more features are matched. Once all retrieved images have been processed this way, we can take those images that have the highest number of matched features to a given u_k and determine how similar those images really are to u_k. Because there may be many false matches, we want to verify that keypoint matches are geometrically consistent. To this end, we attempt to find an affine transformation from u_k to each potential image match. The number of matched features that fit a transformation closely are its consensus set. If two images contain similar objects and scence, then we would expect to find a transform with a large consensus set; if we couldn't find such a transform, then it's unlikely that the two images are related or similar.

4.3.1 Affine Transform for Verification

Affine transforms are linear transforms followed by translation. They preserve the collinearity and the ratio of distances between points on a line. Under affine transform, point $\mathbf{x_1}$ in one image is mapped to a point $\mathbf{x'_1}$ in another image as follows:

$$\begin{bmatrix} x'_1 \\ y'_1 \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{H}} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \tag{4.1}$$

We can rewrite Equation 4.1 so that all the unknown parameters of the transform matrix $\mathbf{H}$ are in one column vector $\mathbf{h}$:

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_1 & y_1 & 0 & 1 \\ x_2 & y_2 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_2 & y_2 & 0 & 1 \\ & & \vdots & & & \\ & & & & & \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} a_{11} \\ a_{12} \\ a_{21} \\ a_{22} \\ t_x \\ t_y \end{bmatrix}}_{\mathbf{h}} = \underbrace{\begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ \end{bmatrix}}_{\mathbf{b}} \tag{4.2}$$

Finding a solution for $\mathbf{h}$ requires a minimum of 3 point-to-point correspondences. The least squares solution is

$$\mathbf{h} = [\mathbf{A}^{\mathrm{T}}\mathbf{A}]^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b} \tag{4.3}$$

Two image can have a large number of putative correspondences, some of which are bad matches/outliers and should not be used to estimate the transform matrix. However, we do not know which correpondences are outliers and which ones are inliers. So the problem is finding both the set of inliers (good correspondences) and an affine transformation simultaneously. This can be done with iterative algorithm RAndom SAmple Consensus (RANSAC) [8].

In each iteration of RANSAC, the algorithm first selects a random sample of s correspondences ($s >= 3$) from all putative correspondences. An affine transform from this sample is calculated according to Equation 4.3. Having obtained a transform, the size of its consensus set can be found by counting number of putative correpondences that fit this transform to within a threshold error. The error of an affine transform for a point $\mathbf{x}$ and its correspondence $\tilde{\mathbf{x}}$ is the transfer error:

$$e = d(\tilde{\mathbf{x}}, \mathbf{H}\mathbf{x})^2$$

The threshold for this error is dependent on application and adjusted empirically.

At the end of each iteration, the fraction of inliers can be updated as necessary based on the maximum size of the consensus set found so far. Knowing the fraction of inliers and given a fixed sample size s, the number of iterations L to run this algorithm is determined to ensure a high probability of generating a good transform. At the end of all iterations, if the largest consensus set is less than a given threshold, then the image match may be rejected as a false match. Otherwise, the transform that produced the biggest consensus set is recalculated using all correspondences in the consensus set to generate the final transform. The similarity between the two images is based in part on the least squared error of this final transform.

4.4 Image Matching with LBP+color features

Matching images with the LBP+color descriptors is much simpler compared to matching with SIFT descriptors. Recall from Section 3.1 that each image is represented as a term-frequency vector of visterms, which come from the k-means clustering of LBP+color histograms. Because there is only one vector per image, no geometric verification is necessary. We can use the same feature indexing structure, the kd-tree, to store the LBP+color visterm vectors. Image matching is the same as feature matching for SIFT, by k-NN search. However, whereas SIFT features are compared using the Euclidean distance between descriptors, LBP+color vectors are compared by their cosine similarity, a measure that is frequently used in text mining to match the tf-idf vectors of documents. The cosine similarity between two LBP+color visterm frequency vectors $\mathbf{a}$ and $\mathbf{b}$ is defined as:

$$\text{sim}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}|| \, ||\mathbf{b}||}$$

where $||\mathbf{a}|| = \sqrt{\mathbf{a} \cdot \mathbf{a}}$. The cosine similarity is basically the cosine of the angle between the vectors $\mathbf{a}$ and $\mathbf{b}$.

Since the tf-idf vectors have no negative-valued components, the cosine similarity takes values in the range $[0, 1]$. A similarity value of 1 means that the two images represented by $\mathbf{a}$ and $\mathbf{b}$ are the most visually similar, while a value of 0 means that they are the least similar.

4.5 Feature Evaluation

We evaluate each stage of our annotation pipeline individually so that it is easier to see where the performance bottleneck is and where we should spend the most effort in making adjustments to the system. At the visual matching stage, there are two important questions we want to answer. First, how good does the matching ability of SIFT compare with that of LBP+color? These are two very different features, and they play a large role in retrieval quality. If we can determine that one feature is significantly better than the other or if both yield similar results, then a lot of effort can be saved by extracting only the better feature. On the other hand, if the features complement each other under different settings, then it is worth our time to investigate a way to combine them.

The second question we want to answer is, how is the retrieval performance of visually similar images influenced by the size of the image features index and the number of noisy/irrelevant images in the index? We would like the matching to be robust to noise, because the initial annotations generated by PAMIR has a low average number of correct annotations per image. This means that the image by text-retrieval stage (stage 2) returns noisy mixtures of relevant and irrelevant images that visual similarity matching must then filter. If matching has a high precision despite the noise, then we do not need to spend too much time improving the previous steps. If this turns out not to be the case, then we need better models for generating the initial annotations.

4.5.1 Metrics

We use *precision* and *recall* to measure the quality of visual similarity matching, and they are defined as follows:

$$\text{precision} = \frac{\text{number of relevant images retrieved}}{\text{total number of images retrieved}}$$

$$\text{recall} = \frac{\text{number of relevant images retrieved}}{\text{total number of relevant images in the index}}$$

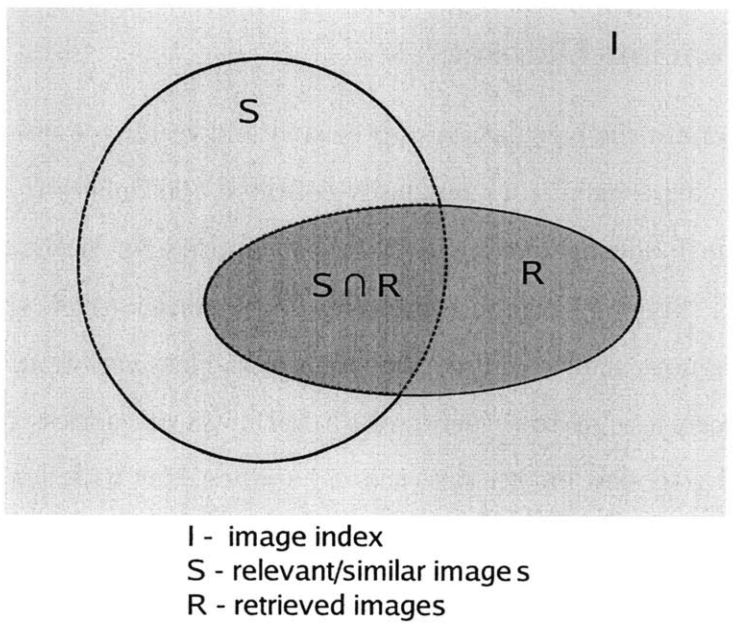


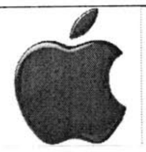
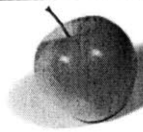
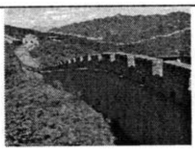
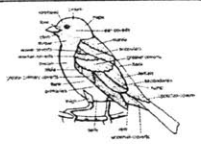
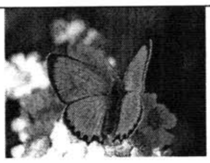
Figure 4-4: A Venn diagram to illustrate precision and recall

These two metrics are very commonly used in information retrieval, and they are appropriate here because visual similarity matching is essentially content-based retrieval. Figure 4-4 gives an example to better illustrate precision and recall. In this example, precision is $\frac{|S \cap R|}{|R|}$, and recall is $\frac{|S \cap R|}{|S|}$. As mentioned before, it is important that the matching has high precision. We would also like the recall rate to be reasonably high, because the next stage gathers the labels of visually similar images to improve the initial annotations, and more similar images yield better text. However, precision and recall often go in opposite directions: when one goes up, the other tend to go down. The higher the precision, the harder it is to find all similar images from a database in the presence of noise, hence the lower the recall and vice versa. For LBP+color feature, we can adjust the values of precision and recall directly by returning more neighbors from the index. Obviously if we returned all images in the index, the recall would be 100% but the precision would then be extremely low. For SIFT features, we cannot control directly the number of similar images retrieved. We can only indirectly influence these values by changing the threshold error of the affine transform step. The higher the allowable error, the higher the recall rate and the lower the precision.

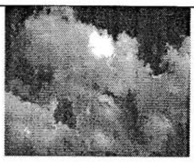
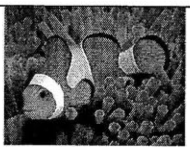
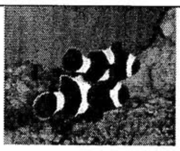
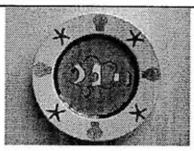
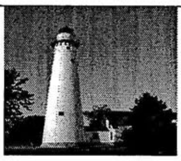
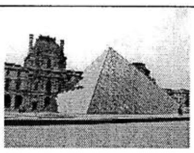
4.5.2 Evaluation Datasets

In order to automate the calculation of precision and recall, we need a ground truth dataset. For our experiments, we manually collected 800 images from Google Image Search using the following queries: apple, beach, Beijing, bird, butterfly, clouds, clownfish, Japan, liberty, lighthouse, Louvre, Paris, sunset, tiger, tree. Many of the queries are ambiguous semantically or has high intra-class variation, so we split these queries into categories. At first there are a total of 38 categories. After eliminating the ones that have too few images (less than 4) and merging back the ones that turned out to be not distinctive enough, 26 categories are left. Table 4.2 shows the queries, the categories, and an example image for each category.

Table 4.2: Examples of ground truth images with queries and categories

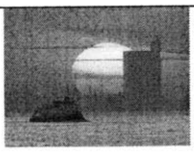
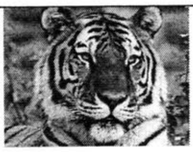
Query	Categories: id, brief description, and example images		
apple	 1. apple0 logo	 2. apple1 fruit	 3. apple2 iphone
beach	 4. beach		
Beijing	 5. beijing0 Tiananmen	 6. beijing1 Great Wall	 7. beijing3 Temple of Heaven
bird	 8. bird0 animal	 9. bird2 diagram	
butterfly	 10.butterfly0		

Continued on next page

Query	Categories: id, description, and example images		
clouds	 11. clouds		
clownfish	 12.clownfish0 orange	 13.clownfish1 black	 14.clownfish2 drawing
Japan	15. japan0 flag	16. japan1 map	
liberty	17.liberty0		
lighthouse	 18.lighthouse		
Louvre	 19. Louvre		

Continued on next page

Table 4.2 – continued from previous page

Query	Categories: id, description, and example images			
Paris	20. paris1 Eiffel Tower	21. paris2 Arch of Triumph	22. paris3 Paris Hilton	23. paris4 map
sunset	 24. sunset0			
tiger	 25. tiger0			
tree	26. tree0			

To study how the size of image index and the percent of irrelevant images affects precision and recall, we test retrieval performance on three datasets of increasing sizes. The first dataset consists of the ground truth images only. This roughly correspond to the situation where initial words predicted by PAMIR is 100% accurate, so retrieval using those initial annotations as keywords should return the fewest number of irrelevant images. The second dataset consist of a mixture of the 800 ground truth images and 10K randomly selected pictures from Google Image Search. This roughly correspond to the situation where 10% of initial annotations are good. The third dataset consist of a mixture of the ground truth with 45K random pictures. The number of images in each dataset and the number of keypoints in the SIFT features index are summarized in Table 4.3.

Table 4.3: Evaluation data statistics

Dataset	no. images	total no. keypoints
dataset1	800	348,000
dataset2	10,800	3,780,000
dataset3	45,800	17,400,000

For each category, we randomly pick a few pictures in the ground truth as query for visual similarity matching against the entire database. Since we know the category that the query image falls into, we can record the number of similar images retrieved that come from the same category in the ground truth. Usually the query image itself is also retrieved but we do not count it when calculating recall and precision.

4.5.3 Results

Table 4.4 shows the matching precision and recall using SIFT feature on the three datasets.

Table 4.4: Precision and recall for SIFT

Dataset	dataset1	dataset2	dataset3
recall	12.6%	11.7%	9.18%
precision	95.3%	94.1%	93.7%

The results confirm that SIFT features are highly distinctive and yield very high matching precision. As we increase the size of the index, both recall and precision decrease as expected. But even on the largest dataset, SIFT does very well. These numbers are obtained by setting the affine transform error threshold to 3 pixels. We also tried setting the threshold to 0.5, 10, and 20 pixels, but setting it to 3 resulted in the best retrieval performance.

Figure 4-5 shows the precision and recall curve on dataset1 using the LBP+color feature for matching. Whereas it is hard to make a curve for SIFT, it is easy to do so for LBP+color because we can directly control the number of similar images retrieved. The curve is created by varying the number of retrieved images from 1 to 12. When only one similar image is returned, the recall is the lowest but the precision is the highest. On the other extreme is when we return 12 images, for which precision is the lowest but recall is highest. The optimal point seems to be when we set the number of images retrieved to 9, for which the precision is 38% and the recall is 17%.

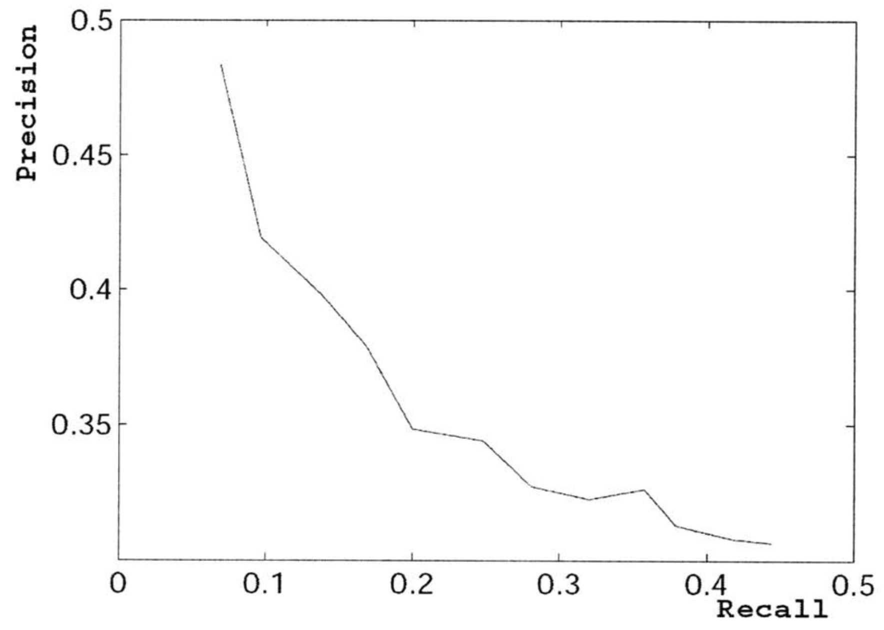


Figure 4-5: Precision-recall curve for matching with LBP+color feature

We can see from Figure 4-5 that at the same recall rate, LBP+color has a much lower precision than SIFT (41% vs. 95%), so SIFT may seem to be the clear winner here. However, in our experiments, we were not able to get more than 13% of recall rate on any dataset when matching with SIFT. Indeed, for many queries such as "clouds" and "sunset", matching with SIFT retrieves no images at all, which is no good if the next stage depends on the visually similar images for annotation refinement. Figure 4-6 shows the side-by-side comparison of the recall rates of SIFT and LBP+color for each category of query image. The numbers are calculated on dataset2. For LBP+color matching, we retrieve 10 images for each query. The category IDs are given in Table 4.2.

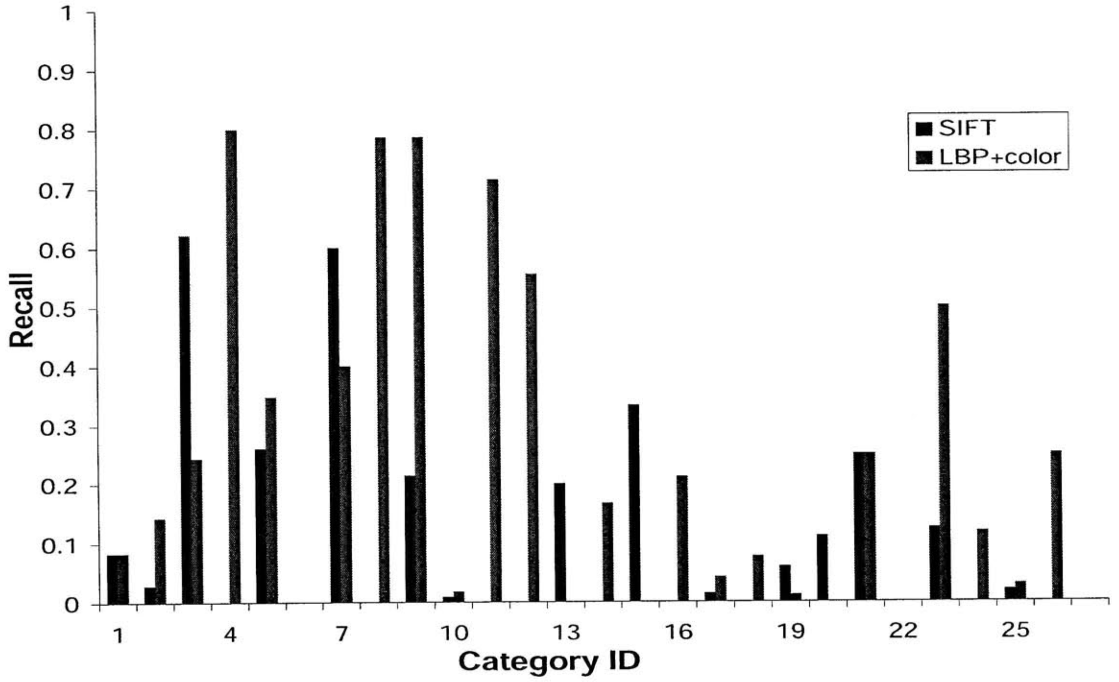


Figure 4-6: Comparison of SIFT and LBP+color recall by category

Clearly, SIFT and LBP+color perform well in different categories. SIFT is better suited to matching rigid shapes. In the ground truth, SIFT has the highest recall in the categories 3-"iphone"(62%) and 7-"Temple of Heaven"(60%). LBP+Color is better for matching images that have nonrigid objects and more color and texture, and it has the highest recall in the categories 4-"beach", 8-"bird", 11-"clouds". These results suggest that a combination of both features may achieve better matching quality. For the rest of the experiments, we use LBP+color for matching, because it is simpler to control the number of similar images retrieved.

Chapter 5

Annotation Refinement

Annotation refinement is the last stage in our pipeline. To recap the previous steps leading to this stage, in the first stage, we predict initial annotations for an unlabeled image using a discriminative model, PAMIR. In the second stage, we use the initial annotations as keywords to retrieve images from Google Image Search. These retrieved images all have some associated text because they have been indexed for search. The third stage involves filtering the retrieved images to retain only the ones that are visually similar to the unlabeled image. Now in this last stage, the retained images, their similarity scores to the unlabeled image, and their associated text are processed to refine the initial annotations. There are two areas for refinement: 1) we can expand the initial set of PAMIR annotations by transfering labels from similar images. 2) we can re-rank the initial annotations so that correct annotations come before incorrect ones. In this chapter, we investigate both areas for improvement, and present some preliminary experimental results.

There are two sources of text associated with images retrieved from Google Image Search: 1) surrounding text and 2) query log. The surrounding text consist of words shown on the same web page as the image. These words are assigned weights when the image is first crawled and added to Google's search index. The weights are either assigned by heuristics or by a fitted model, and are mainly dependent on the word's physical distance to the image in the web page's layout. The higher the weight of a word, the more relevant the word is to the image. The second source of text, the query log, contains information such as which queries returned a particular image, and the fraction of clicks that image got under each query. The higher the click fraction, the more relevant the query words are to the image. Both query log words and surrounding text of an image to the unlabeled image are examined for ways to improve the initial annotations.

5.1 Expansion of Initial Annotations

To expand the initial annotations, we transfer the labels from visually similar images to the unlabeled image. The transferred labels may be ranked by voting. The simplest voting scheme is one where each retrieved image gets equal vote. If m of the retrieved images have the word l in their surrounding text, then the label l gets m votes. However, this voting scheme does not take advantage of all the information we have about the retrieved images. For example, each retrieved image has a matching score that indicates how similar it is to the unlabeled picture. For LBP+color, this matching score is the cosine distance between two visterm-frequency vectors. For SIFT, the matching score depends on the number of good keypoint matches. In both cases, the higher the score, the more similar a retrieved image is to the unlabeled image. Naturally, the labels transferred from the most similar images should have a bigger weight in voting than those taken from the least similar images. Also, each word in the text of retrieved images has its own weight (click fraction or layout distance), that indicates how relevant the label is to the retrieved image. So more relevant labels should also have bigger weights in voting. We choose to weight the votes for a transferred label l by the product of the retrieved image's similarity score and l's original weight. For example, if a retrieved image r has similarity score s, and a word l in its surrounding text has a weight of t, then r's vote for the word l is st. We sum the votes for each transferred label from all retrieved images and rank them by the total votes, and keep only the top 20 words as added annotations. Because the query log and surrounding text words have very different weights, we kept the labels from these two sources separated. Figure 5-1 shows the results of label transfer.

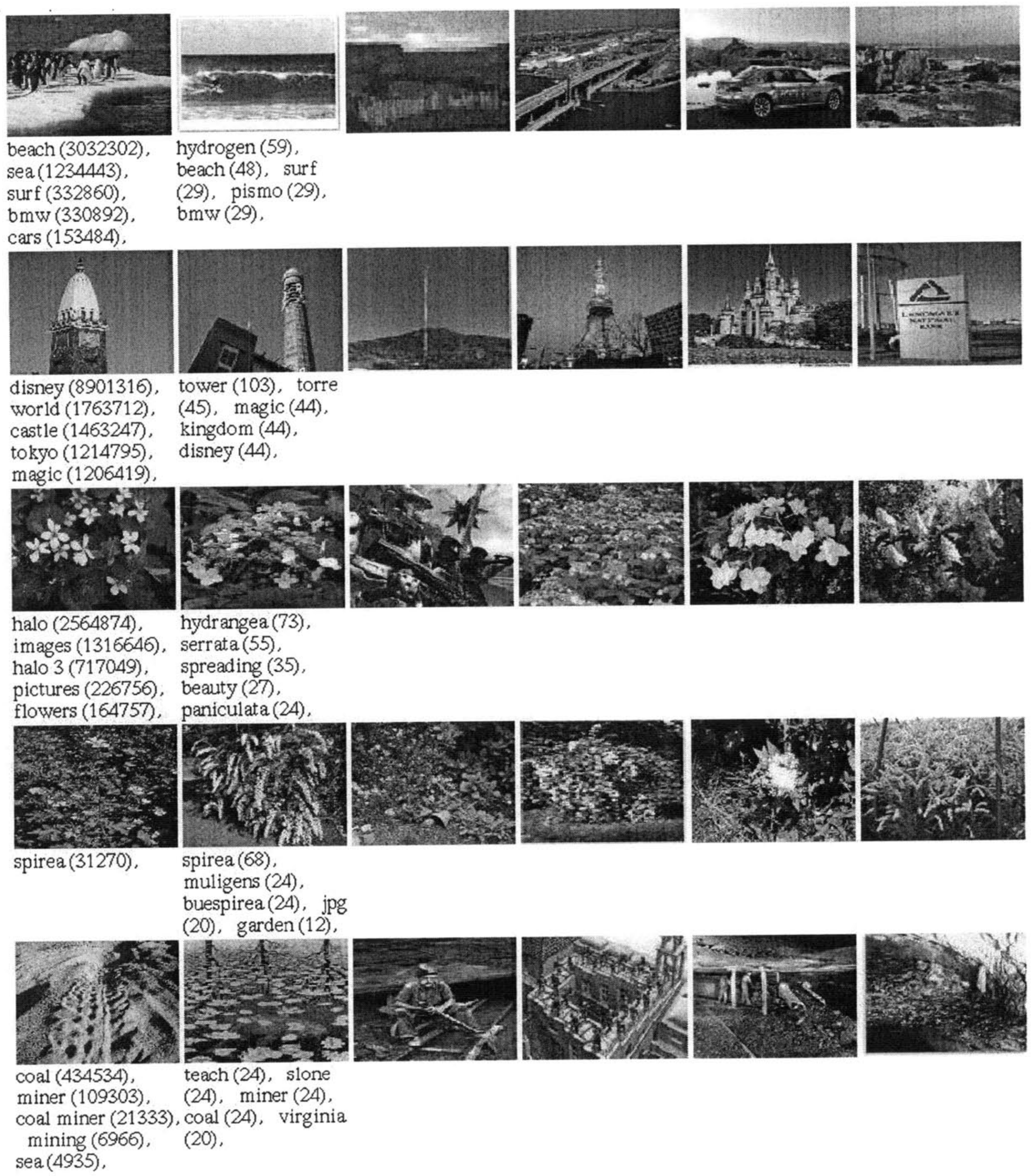

beach (3032302), sea (1234443), surf (332860), bmw (330892), cars (153484),
hydrogen (59), beach (48), surf (29), pismo (29), bmw (29),
disney (8901316), world (1763712), castle (1463247), tokyo (1214795), magic (1206419),
tower (103), torre (45), magic (44), kingdom (44), disney (44),
halo (2564874), images (1316646), halo 3 (717049), pictures (226756), flowers (164757),
hydrangea (73), serrata (55), spreading (35), beauty (27), paniculata (24),
spirea (31270),
spirea (68), muligens (24), buespirea (24), jpg (20), garden (12),
coal (434534), miner (109303), coal miner (21333), mining (6966), sea (4935),
teach (24), slone (24), miner (24), coal (24), virginia (20),

Figure 5-1: Label transfer results: the unlabeled images are shown in the first column. The retrieved images are shown in the rest of the columns in the same row. Transferred labels from the query log are shown in the first column under the unlabeled image, and the labels from the surrounding text of similar images are shown in the second column. The numbers inside the parenthesis are the sum of weighted votes rounded to the nearest integer.

5.2 Re-ranking of Initial Annotations

Instead of transfering *all* labels from similar images to the unlabeled image, we can use the labels from similar images to promote the ranking of that label in the initial annotations. On the Image Labeler dataset, we retrieve 30 visually similar images for each unlabeled image. Labels from similar images are ignored if they do not occur in the initial annotations at all. If they do occur in the initial annotations, the number of similar images containing that word is used as the vote for that word. Clearly, re-ranking of initial annotations alone will not introduce more correct annotations, and so will not improve the coverage rate or accuracy of the initial annotations. However, it does improve the quality of annotation presented to the user by pushing more relevant words to the top of the list.

Figure 5-2 shows the cumulative number of correct annotations at each ranking position in the list of annotations output by the system. PAMIR produced a total of 1871 correct initial annotations for the Image Labeler dataset (recall that we choose the top 11 words for each picture as initial annotations and rank these words by there PAMIR score, which is the dot product of the model W and the visterm vector $\mathbf{p}$). The straight line in Figure 5-2 represents the situation where the correct annotations are randomly and uniformly distributed across all positions in the ranking, so that at each position, we would expect to see $1871/11 = 170$ correct labels generated. In this situation, there would be a cumulative total of $170r$ correct labels at the position r in the ranking. The dashed line plots the cumulative total correct annotations for the initial PAMIR output. Compared to the random case, PAMIR output ranks more correct words in earlier positions. The top line is the curve of the cumulative total after re-ranking, which shows an improvement over PAMIR's initial annotations, in the ranking of correct annotations in earlier positions.

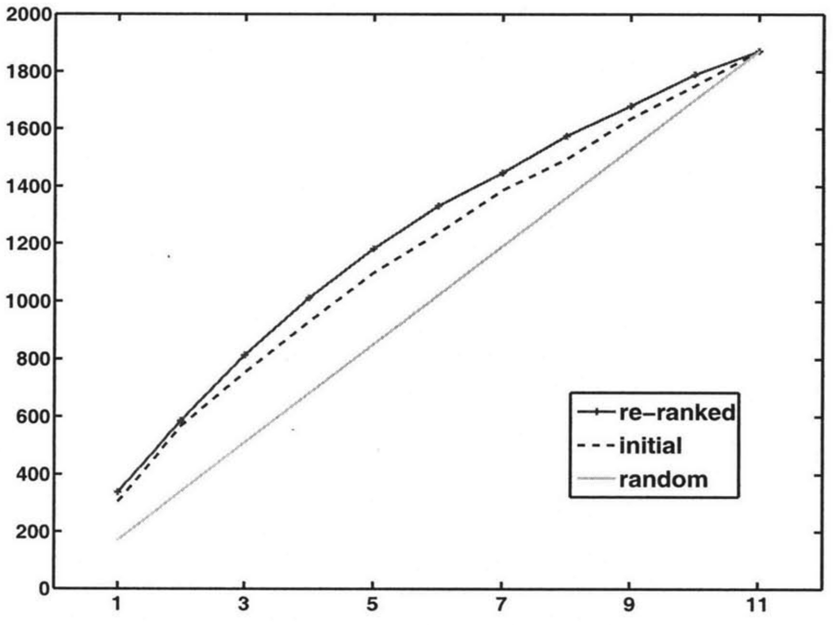


Figure 5-2: Re-ranking results

Chapter 6

Conclusions and Future Work

In this thesis, I have presented the a pipeline for image annotation, which relies a discriminative model for annotation and content-based retrieval for annotation refinement. I have described how to put the various stages together and how to tune and evaluate the effectiveness of each stage. This project is still in early exploratory phase and a lot of additional work can be done to improve it further. This chapter lists many directions for future work and concludes this thesis.

6.1 Adapting PAMIR to Annotation

As the Chapter 3 discussed, the learning goal of PAMIR is to find a classifier that ranks relevant pictures higher than irrelevant pictures given a text query. The goal of annotation, on the other hand, is to rank words given a picture, so that relevant words score higher than irrelevant words. While the two goals are very similar, the model trained for the first goal is not optimized for the second goal. However, we can retrain a model optimized for the second goal by simply tweaking the input to PAMIR's training algorithm: swapping the roles of query and picture vectors, or transposing the relevance matrix R will do the trick. The new training instances would be triples $(\mathbf{p}, \mathbf{q}^+, \mathbf{q}^-)$, and the update equations would remain the same, except Equation 3.6 should be changed to $v_i = \gamma(\mathbf{p}_i, \Delta\mathbf{q}_i)$, where $\gamma(\mathbf{p}, \mathbf{q}) = [p_1\mathbf{q} \quad p_2\mathbf{q} \quad \cdots \quad p_C\mathbf{q}]$.

The model re-trained from the new inputs would optimize the margin between relevant annotation and irrelevant annotation given each image, and should therefore generate better initial annotations.

In addition to training the model with a new goal, it would also be helpful to clean the vocabulary of the training corpus. PAMIR's vocabulary is derived from actual user search queries. However, these may be too diverse and too specific for annotation purposes. For example, it would be better to change the query "hydrangea paniculata" to simply "flower" or "plant". We can eliminate rare words with WordNet, for example by going up one level of definition for more generality.

6.2 Improvements to Image Matching

While model building can to a certain extent make up for the lack of high quality image features, retrieval performance will always benefit directly from the use of better features. More research in this area is well worth the effort. Without groundbreaking new features, a simple way to improve existing features is to combine them. As noted in Section 4.5.3, a simple weighting scheme may be enough to get improved retrieval performance. A more systematice way for combining multiple features was proposed in [10], which learns a distance function for comparing features combinations.

In addition to feature improvement, we also need to evaluate retrieval with a standard dataset. The ground truth images we collected is not a benchmark. Therefore, we cannot really compare our results against other researchers' results. Our dataset may be too small and two easy. A benchmark set such as the Caltech101 or the LabelMe dataset should be used in the future.

6.3 Fusion of transfered labels

Because we use two sources of text for label transfer, one from query logs, and one from surrounding text, it would be desirable to have a strategy to effectively combine both sources of text and rank them together, despite the differences in the properties of these two sources of text. A possible strategy is to assign weights to each source heuristically: a higher weight should probably be given to query log words than surrounding text, since the query click data is harder to manipulate. Ultimately, the optimal weight assignment function should be learned by training from large amounts of data.

6.4 Conclusion

Finally, there have been many methods now for processing images, learning patterns, classifying data. In designing an annotation system such as this one, which combines so many methods from diverse fields, one has to make many choices which are hopefully guided by an analysis of the trade-offs. In this project, the model chosen for annotation and the features used for retrieval has been mainly determined based on the availability of good exisiting implementations and the current trends in this area which may not be long-lasting. Therefore, it would be desirable to take a more principled approach for choosing which methods to use in each stage.

Bibliography

[1] Kobus Barnard, Pinar Duygulu, David Forsyth, Nando De Freitas, David M. Blei, and Michael I. Jordan. Matching words and pictures. *Journal of Machine Learning Research*, 3:1107–1135, 2003.

[2] Jeffrey S. Beis and David G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1000, 1997.

[3] David M. Blei and Michael I. Jordan. Modeling annotated data. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 127–134, New York, NY, USA, 2003. ACM.

[4] Edward Y. Chang, King-Shy Goh, G. Sychay, and Gang Wu. CBSA: content-based soft annotation for multimodal image retrieval using Bayes point machines. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(1):26–38, Jan 2003.

[5] P. Duygulu, Kobus Barnard, J. F. G. de Freitas, and David A. Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part IV*, pages 97–112, London, UK, 2002. Springer-Verlag.

[6] C F. Estrada, A. Jepson, and D. Fleet. Local features tutorial, November 2004.

[7] S.L. Feng, R. Manmatha, and V. Lavrenko. Multiple Bernoulli relevance models for image and video annotation. *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, 2:II–1002–II–1009 Vol.2, June-2 July 2004.

[8] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[9] D. A. Forsyth and M. M. Fleck. Body plans. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 678, Washington, DC, USA, 1997. IEEE Computer Society.

[10] Andrea Frome, Yoram Singer, and Jitendra Malik. Image retrieval and classification using local distance functions. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 417–424. MIT Press, Cambridge, MA, 2007.

[11] King-Shy Goh, Edward Y. Chang, and Kwang-Ting Cheng. SVM binary classifier ensembles for image classification. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 395–402, New York, NY, USA, 2001. ACM.

[12] D. Grangier and S. Bengio. A discriminative kernel-based approach to rank images from text queries. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(8):1371–1384, Aug. 2008.

[13] J. Jeon, V. Lavrenko, and R. Manmatha. Automatic image annotation and retrieval using cross-media relevance models. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 119–126, New York, NY, USA, 2003. ACM.

[14] Thorsten Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, New York, NY, USA, 2002. ACM.

[15] V. Lavrenko, R. Manmatha, and J. Jeon. A model for learning the semantics of pictures. In *Advances in Neural Information Processing Systems 15*, 2003.

[16] Jia Li and James Z. Wang. Automatic linguistic indexing of pictures by a statistical modeling approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(9):1075–1088, 2003.

[17] Jiakai Liu, Rong Hu, Meihong Wang, Yi Wang, and Edward Chang. Web-scale image annotation. In *Pacific-Rim Conference on Multimedia 2008*, pages 663–674, Berlin, Germany, 2008. Springer.

[18] Jing Liu, Bin Wang, Mingjing Li, Zhiwei Li, Weiying Ma, Hanqing Lu, and Songde Ma. Dual cross-media relevance model for image annotation. In *MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia*, pages 605–614, New York, NY, USA, 2007. ACM.

[19] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[20] Florent Monay and Daniel Gatica-Perez. On image auto-annotation with latent space models. In *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, pages 275–278, New York, NY, USA, 2003. ACM.

[21] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):971–987, 2002.

[22] Edgar Osuna, Robert Freund, and Federico Girosi. Training Support Vector Machines: an Application to Face Detection. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 130, Washington, DC, USA, 1997. IEEE Computer Society.

[23] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(12):1349–1380, Dec 2000.

[24] A. Torralba, R. Fergus, and W.T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(11):1958–1970, Nov. 2008.

[25] A. Vailaya, A. Jain, and H. J. Zhang. On image classification: City vs. landscape. In *CBAIVL '98: Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries*, page 3, Washington, DC, USA, 1998. IEEE Computer Society.

[26] Luis von Ahn and Laura Dabbish. Labeling images with a computer game. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326, New York, NY, USA, 2004. ACM.

[27] J.Z. Wang, Jia Li, and G. Wiederhold. Simplicity: semantics-sensitive integrated matching for picture libraries. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(9):947–963, Sep 2001.

[28] Xin-Jing Wang, Lei Zhang, Xirong Li, and Wei-Ying Ma. Annotating images by mining image search results. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(11):1919–1932, Nov. 2008.

[29] T. Yeh, K. Tollmar, and T. Darrell. Searching the web with mobile images for location recognition. *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, 2(1):II–76–II–81 Vol.2, June-2 July 2004.